Suborbital ❤️ WasmEdge

Michael Yuan

Dec 20, 2021 · 3 min read

With contribution from Shishuo Wang

The WasmEdge team is happy to announce that WasmEdge Runtime is now fully supported in Suborbital's Reactr function scheduler as an embedded WebAssembly engine!

Reactr is a Go-based multi-tenant function scheduling and management framework. It allows WebAssembly functions, compiled from languages such as Rust, AssemblyScript, Swift, TinyGo, Grain, to be embedded in Go applications. It can manage and execute hundreds of WebAssembly function instances all at once. That makes Reactr very useful in SaaS applications, where users could embed their own functions and logic to customize the SaaS for their own use cases.

Besides multi-language support, the key benefits of using WebAssembly as a Go extension / plug-in mechanism include small footprint, high performance, memory safety, sandbox security, and cross platform compatibility. Furthermore, Reactr extends the standard WebAssembly spec to allow embedded WebAssembly functions access to many important capabilities implemented in native Go. It does so by registering a set of Go-based host functions to the embedded WebAssembly runtime, and then provides Rust / AssemblyScript / Swift API libraries for developers to call those Go host functions from the WebAssembly sandbox. For example, Reactr WebAssembly functions can make HTTP requests or even query relational databases via these capability APIs. That nicely supplements the core WebAssembly features the embedded runtime provides.

Reactr aims to support multiple embedded WebAssembly runtimes since each runtime has its own performance characteristics, extensions, use cases, and community. WasmEdge is an open source WebAssembly runtime designed and optimized for cloud-native and edge-native application use cases. It is the only WebAssembly runtime project hosted by the CNCF (Cloud native computing foundation) and Linux Foundation.

WasmEdge is one of the fastest WebAssembly runtimes on the market. In Reactr's official test suites, WasmEdge runs about 30% to 50% faster than other WebAssembly runtimes Reactr supports.

WasmEdge also supports its own WebAssembly extensions. For example, WasmEdge provides extensions and APIs to access network sockets, native AI inference libraries and hardware (eg Tensorflow), and key value stores. WasmEdge also provides advanced JavaScript support, including support for ES6 modules, CommonJS modules, NPM modules, HTTP networking, Tensorflow inference and even Rust / JavaScript mixed functions. Through Reactr and WasmEdge integration, developers now have a safe and efficient mechanism to embed JavaScript function into Go applications.

Finally, since WasmEdge is written in portable C++, it can be compiled to a variety of OS and hardware platforms, and can link to a large number of shared libraries (e.g., the above mentioned Tensorflow).

To use WasmEdge with Reactr is very easy. You just need to install WasmEdge via a one-line installer, and then append the `--tags wasmedge` option to your Go command when compiling your Reactr application. The Reactr GitHub actions provide a great example in how to do this.

To learn more about how to run WasmEdge applications in Reactr, including how to embed complex JavaScript programs in Reactr, you can check out the tutorials in WasmEdge documentation.

- Prerequisites
- Hello world
- Database query
- Embedded JavaScript

Happy coding!

WebAssembly · Go Language · Rust · performance

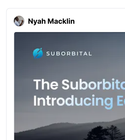
MORE ARTICLES

Ramón Huidobro

The Road Ahead: Insights from Wasm I/O on the Future of Web Assembly

The inaugural Wasm I/O conference took place last month, bringing together industry leaders and expe...

Nyah Macklin



The Suborbital Editor: Introducing Editor V2

Houston, we are officially in orbit. The SE2 Plugin Editor: The Suborbital Extension Engine (SE2) Pl...

Dan Callahan

Bringing Python to SE2 with WebAssembly

Since launching the Suborbital Extension Engine (SE2), support for Python has been one of our top fe...